US010541939B2

(12) United States Patent
Duarte et al.

(10) Patent No.: US 10,541,939 B2
(45) Date of Patent: Jan. 21, 2020

(54) SYSTEMS AND METHODS FOR PROVISION OF A GUARANTEED BATCH

(71) Applicant: **Google LLC**, Mountain View, CA (US)

(72) Inventors: **Alexandre Duarte**, Milpitas, CA (US); **Yingchong Situ**, Campbell, CA (US); **Robert van Gent**, Redwood City, CA (US); **Walfredo Cirne Filho**, Palo Alto, CA (US); **Ramy Abdelaal**, Waterloo (CA); **Smeeta Jalan**, San Jose, CA (US); **Maya Haridasan**, San Jose, CA (US)

(73) Assignee: **Google LLC**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 284 days.

(21) Appl. No.: **15/677,916**

(22) Filed: **Aug. 15, 2017**

(65) **Prior Publication Data**

US 2019/0058669 A1 Feb. 21, 2019

(51) **Int. Cl.**

H04L 12/24	(2006.01)
H04L 12/911	(2013.01)
G06F 9/50	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 47/78*** (2013.01); ***G06F 9/5005*** (2013.01); ***H04L 41/14*** (2013.01); ***H04L 41/147*** (2013.01);

(Continued)

(58) **Field of Classification Search**

CPC H04L 12/00; H04L 12/14; H04L 12/1432–1439; H04L 41/00; H04L 41/08; H04L 41/0803; H04L 41/0813; H04L 41/0816; H04L 41/085; H04L 41/14–147; H04L 41/50; H04L 41/5009–5016; H04L 41/5038; H04L 47/00; H04L 47/50;

(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,785,889 B1 * 8/2004 Williams G06F 9/50 709/229

8,429,630 B2 4/2013 Nickolov et al.

(Continued)

FOREIGN PATENT DOCUMENTS

WO 2016020731 A1 2/2016

OTHER PUBLICATIONS

Babaioff et al., "ERA: A Framework for Economic Resource Allocation fot the Cloud" Feb. 24, 2017, Microsoft.*

(Continued)

Primary Examiner — Michael A Keller

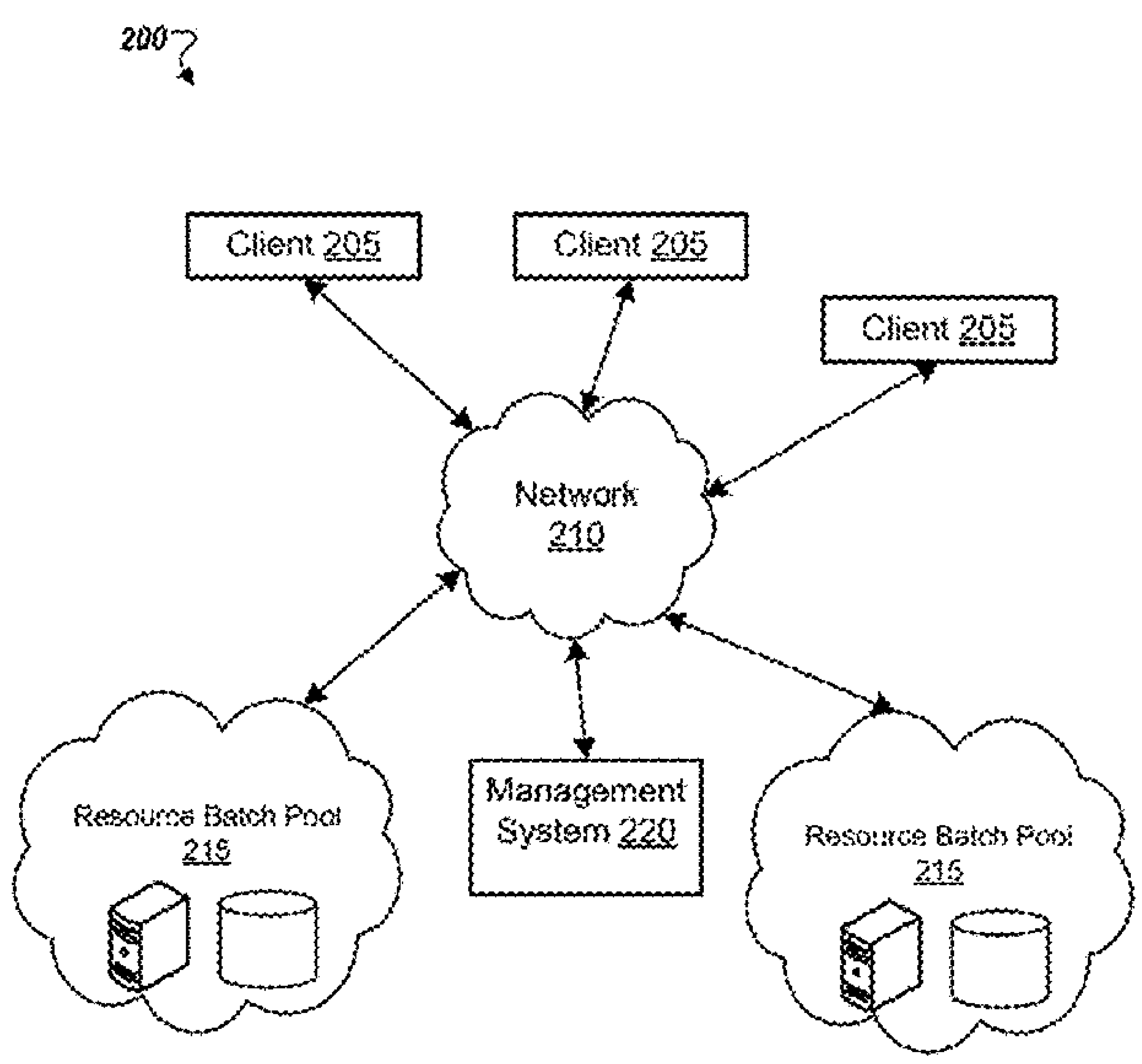
Assistant Examiner — Boris D Gridjalva Lobos

(74) *Attorney, Agent, or Firm* — Honigman LLP; Brett A. Krueger

(57) **ABSTRACT**

Systems and methods for providing a guaranteed batch pool are described, including receiving a job request for execution on the pool of resources; determining an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources; determining a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time; determining that the job request is capable of being executed for the amount of time; and executing the job request over the amount of time, according to the resource allocation.

15 Claims, 10 Drawing Sheets



(52) **U.S. Cl.**
CPC ***H04L 41/50*** (2013.01); ***H04L 41/5009*** (2013.01); ***H04L 41/5038*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 47/56; H04L 47/468; H04L 47/70; H04L 47/78; G06F 9/46; G06F 9/50; G06F 9/5005; G06F 9/5027; G06F 9/5038; G06F 9/5061; G06F 2209/5011
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,583,799 B2 * 11/2013 Podila G06F 9/5011 709/224
8,621,476 B2 * 12/2013 Youn G06F 9/5061 718/104
8,755,837 B2 6/2014 Rhoads et al.
8,832,219 B2 9/2014 Morgan
8,997,109 B2 * 3/2015 Lee G06F 9/505 718/105
9,009,316 B2 4/2015 Tontiruttananon et al.
9,124,650 B2 9/2015 Maharajh et al.
9,294,236 B1 * 3/2016 Ward, Jr. H04L 1/20
9,336,042 B1 * 5/2016 Brenneman H04L 41/0813
9,378,065 B2 6/2016 Shear et al.
9,417,919 B2 8/2016 Simitsis et al.
9,578,088 B2 2/2017 Nickolov et al.
9,612,878 B2 * 4/2017 Di Balsamo G06F 9/5083
9,886,311 B2 * 2/2018 Miller G06F 11/3024
10,152,449 B1 * 12/2018 Ward, Jr. G06F 9/50
10,218,639 B2 * 2/2019 Kandula H04L 67/1095
2005/0207439 A1 * 9/2005 Iyengar H04L 67/322 370/428
2005/0268299 A1 * 12/2005 Picinich G06F 9/485 718/100
2007/0240161 A1 * 10/2007 Prabhakar G06F 9/5044 718/104
2008/0008085 A1 * 1/2008 Gerstel H04L 12/14 370/216
2008/0115143 A1 * 5/2008 Shimizu G06F 9/5066 718/105
2008/0316938 A1 12/2008 Shi
2009/0172655 A1 * 7/2009 Ivanov G06F 8/65 717/173
2011/0154353 A1 * 6/2011 Theroux G06F 9/5038 718/104
2012/0131591 A1 * 5/2012 Moorthi G06Q 10/06 718/104
2012/0173299 A1 * 7/2012 McMullin G06Q 10/06 705/7.26
2012/0284077 A1 * 11/2012 Xiao G06Q 10/20 705/7.15
2013/0091282 A1 4/2013 Tontiruttananon et al.
2013/0117454 A1 * 5/2013 Rangarajan G06F 9/5011 709/226
2013/0179371 A1 * 7/2013 Jain G06F 9/5027 705/400
2014/0025796 A1 * 1/2014 Vibhor H04L 67/1097 709/222
2014/0075222 A1 * 3/2014 Jackson G06F 1/329 713/320
2015/0039552 A1 * 2/2015 Moyne G06Q 40/00 706/52
2015/0149999 A1 * 5/2015 Ramanathan G06F 9/4856 718/1
2015/0150016 A1 * 5/2015 Kim G06F 9/4881 718/103
2015/0264135 A1 * 9/2015 Kandula H04L 67/1095 709/204
2015/0331715 A1 * 11/2015 Sathyanarayana H04L 67/10 709/226
2017/0300364 A1 * 10/2017 Liu H04W 4/029
2019/0058669 A1 * 2/2019 Duarte G06F 9/5005

OTHER PUBLICATIONS

Hoffmann, H., et al., Dynamic Knobs for Responsive Power-Aware Computing, ASPLOS'11, Mar. 5-11, 2011, Newport Beach, CA, 14 pgs.
Hall, J. E., Distributed Control System for Demand Response by Servers, Master of Science Thesis, University of Iowa, 2015, 59 pgs.
Amazon.com, What is Cloud Computing?, 67 pgs., Dec. 1, 2014, accessed via https://www.slideshare.net/AmazonWebServices/what-is-cloud-computing-42208059.
Microsoft.com, Azure Resource Manager Overview, Aug. 10, 2017, 9 pgs.

* cited by examiner

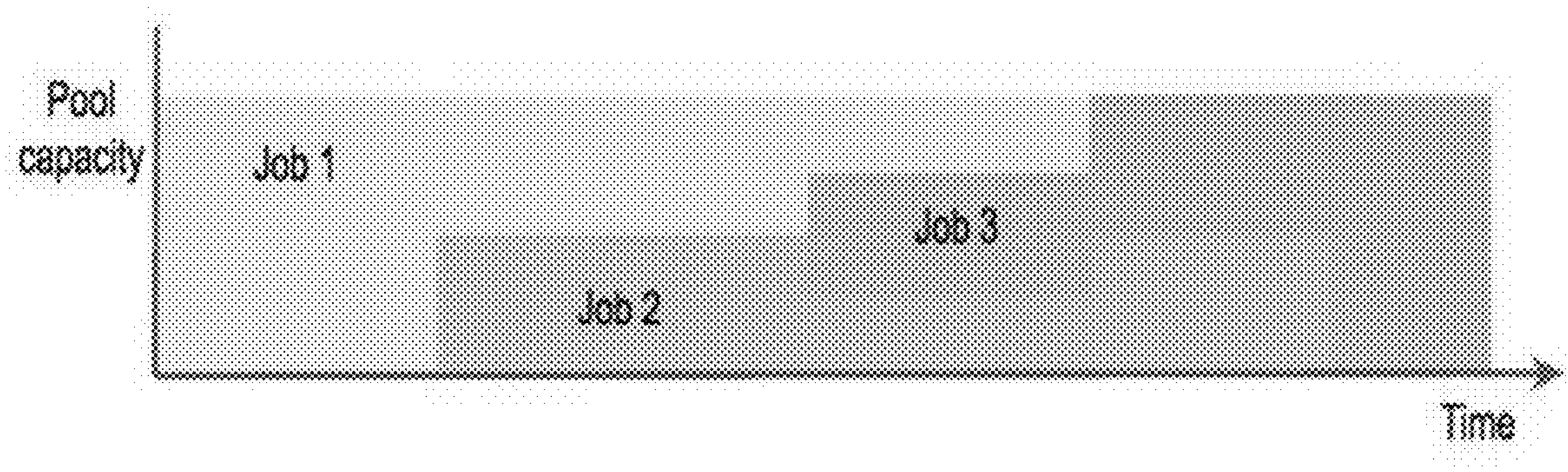

Pool capacity
Job 1
Job 2
Job 3
Time

FIG. 1(a)

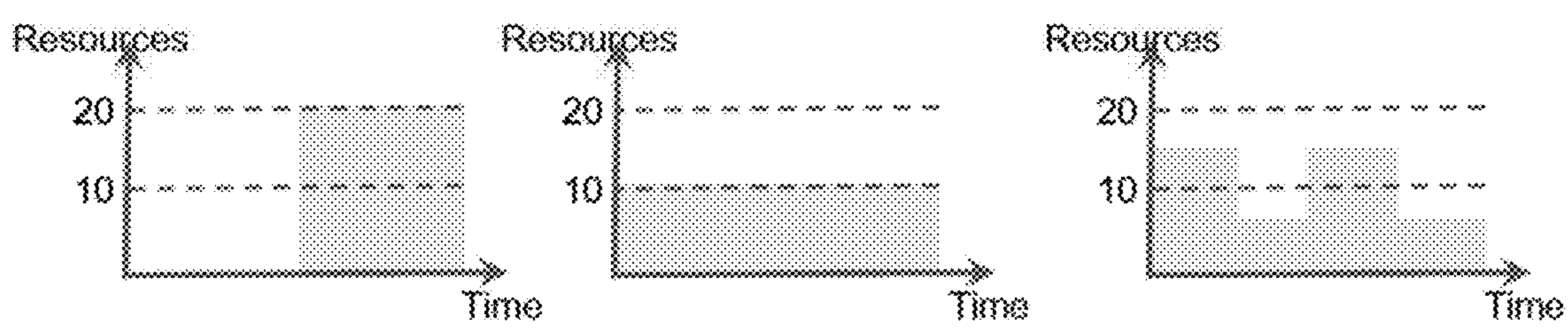

Resources
20
10
Time
Resources
20
10
Time
Resources
20
10
Time

FIG. 1(b)

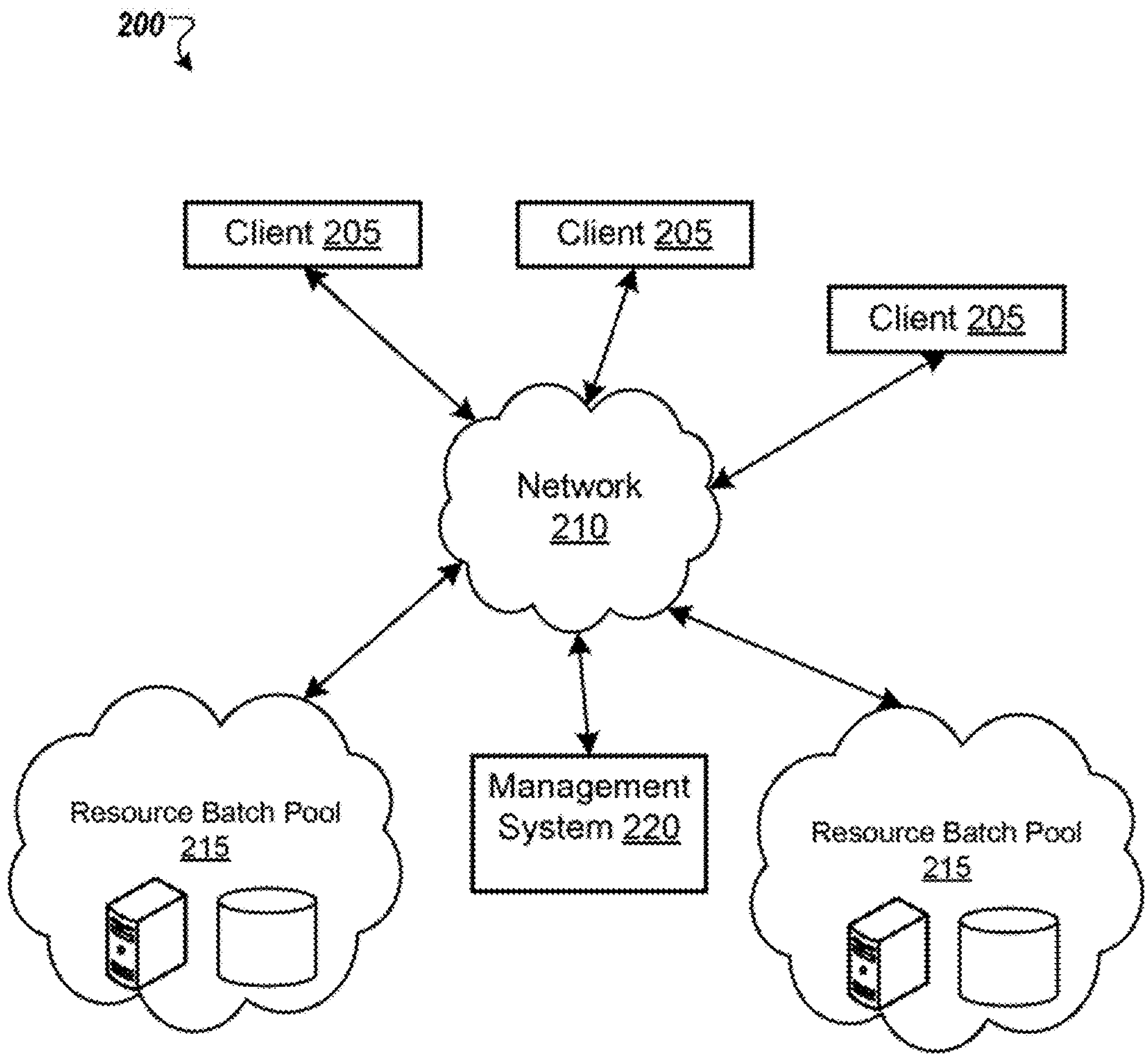

200
Client 205
Client 205
Client 205
Network 210
Resource Batch Pool 215
Management System 220
Resource Batch Pool 215

FIG. 2

300	301	302	303	304
User ID	Budget	Commitments	Historical	Job Requests
USER1	BUD1	COM1	HIS1	JOB1, JOB2, JOB3...
USER2	BUD2	COM2	HIS2	NONE
USER3	BUD3	COM3	HIS3	JOB4, JOB5....
....	....	....	....	....

FIG. 3(a)

310	311	312	313	314
Resource Type	Time Period	Commitments	Historical	Expected
RTYPE1	TIM1	RCOM1	RHIS1	EXP1
RTYPE2	TIM2	RCOM2	RHIS2	EXP2
RTYPE3	TIM3	RCOM3	RHIS3	EXP3
....	....	....	....	....

FIG. 3(b)

304-1 304-2 304-3 304-4

Job Request ID	Tasks	Resource allocation	Constraints
JOB1	TASK1, TASK2, ...	RES1	CON1
JOB2	TASK3	RES2	CON2
JOB3	TASK4, TASK5...	RES3	CON3
...	...	...	...

FIG. 3(c)

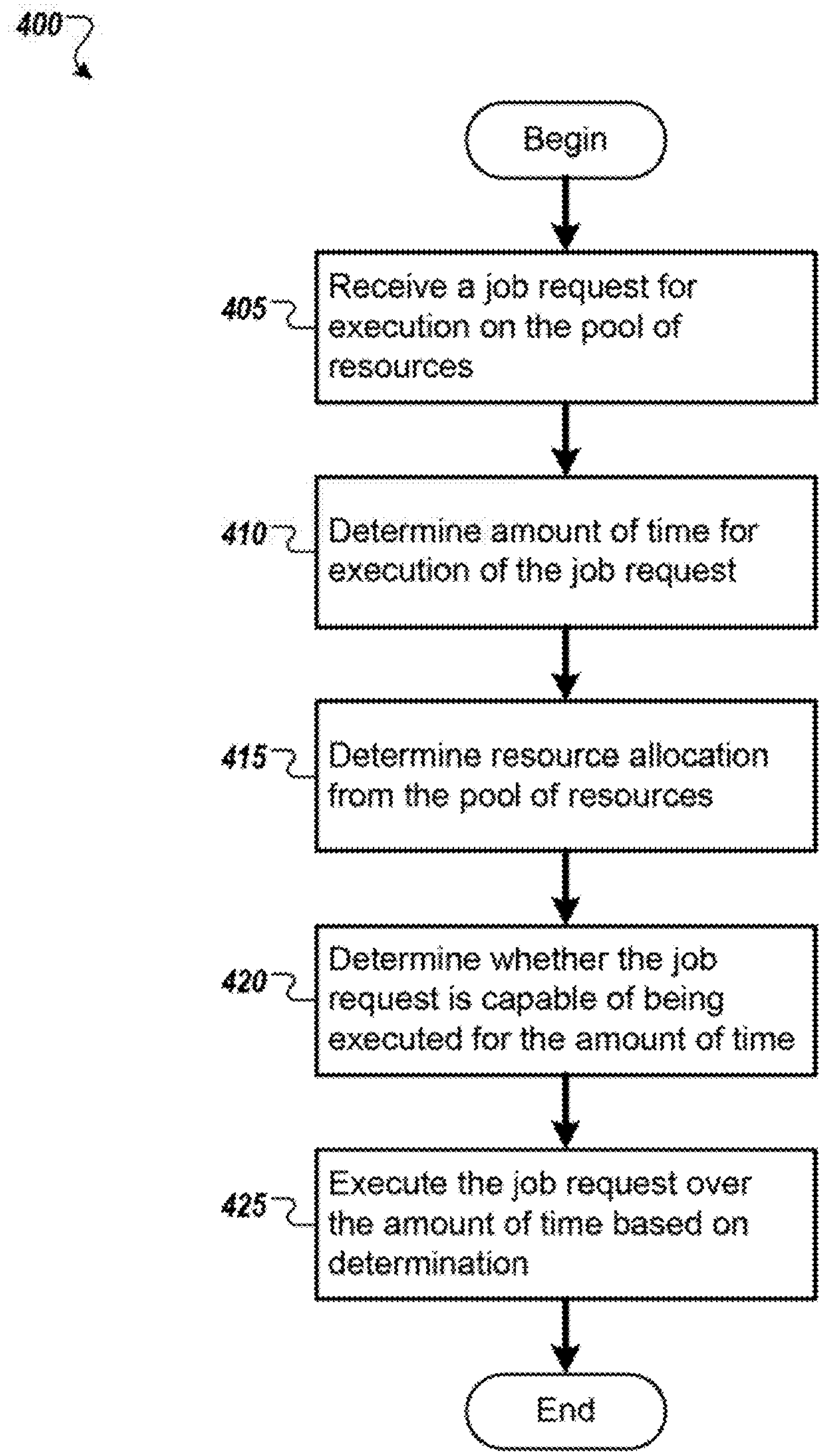

400
Begin
405
Receive a job request for execution on the pool of resources
410
Determine amount of time for execution of the job request
415
Determine resource allocation from the pool of resources
420
Determine whether the job request is capable of being executed for the amount of time
425
Execute the job request over the amount of time based on determination
End

FIG. 4(a)

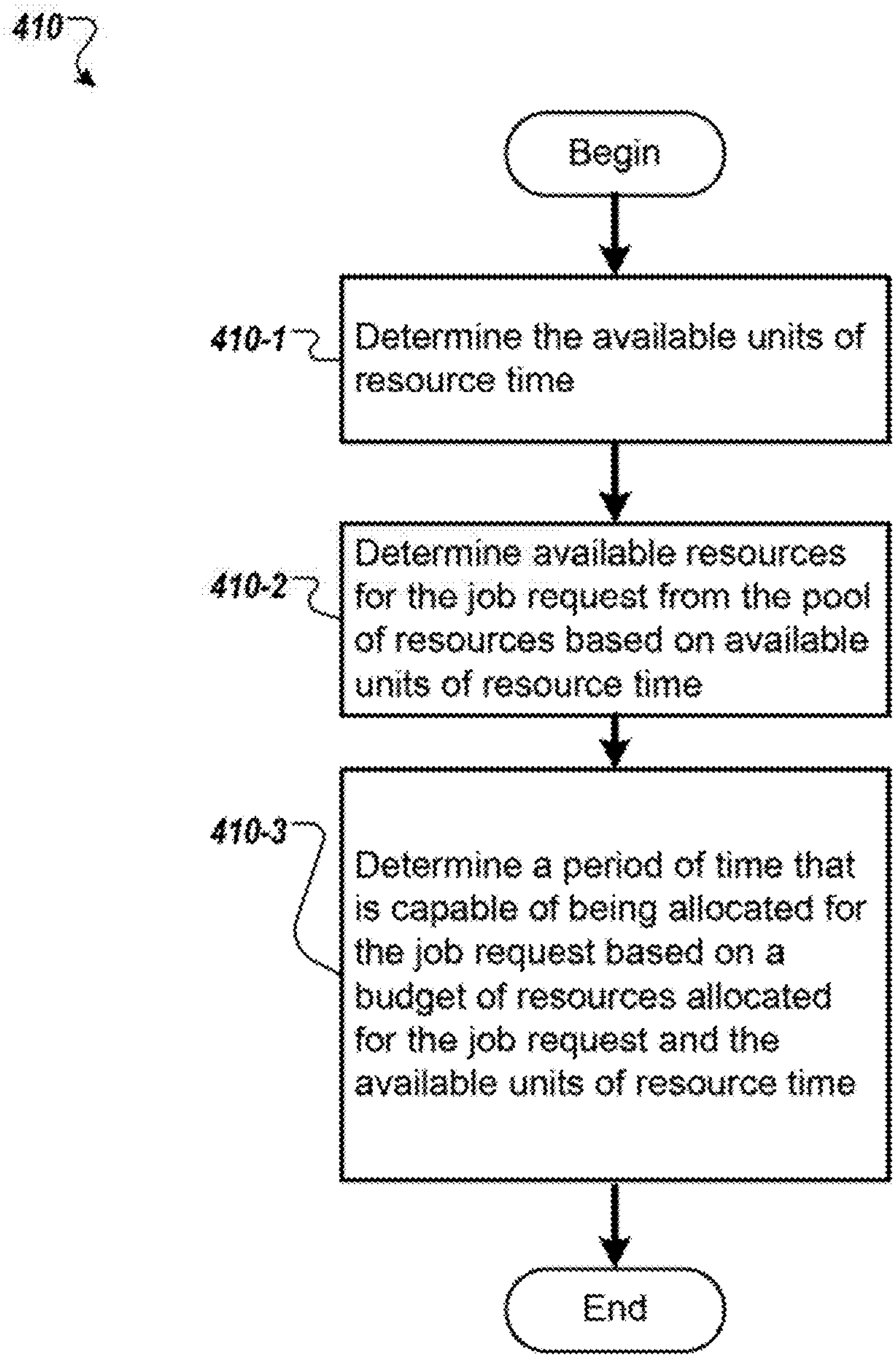

410
Begin
410-1
Determine the available units of resource time
410-2
Determine available resources for the job request from the pool of resources based on available units of resource time
410-3
Determine a period of time that is capable of being allocated for the job request based on a budget of resources allocated for the job request and the available units of resource time
End

FIG. 4(b)

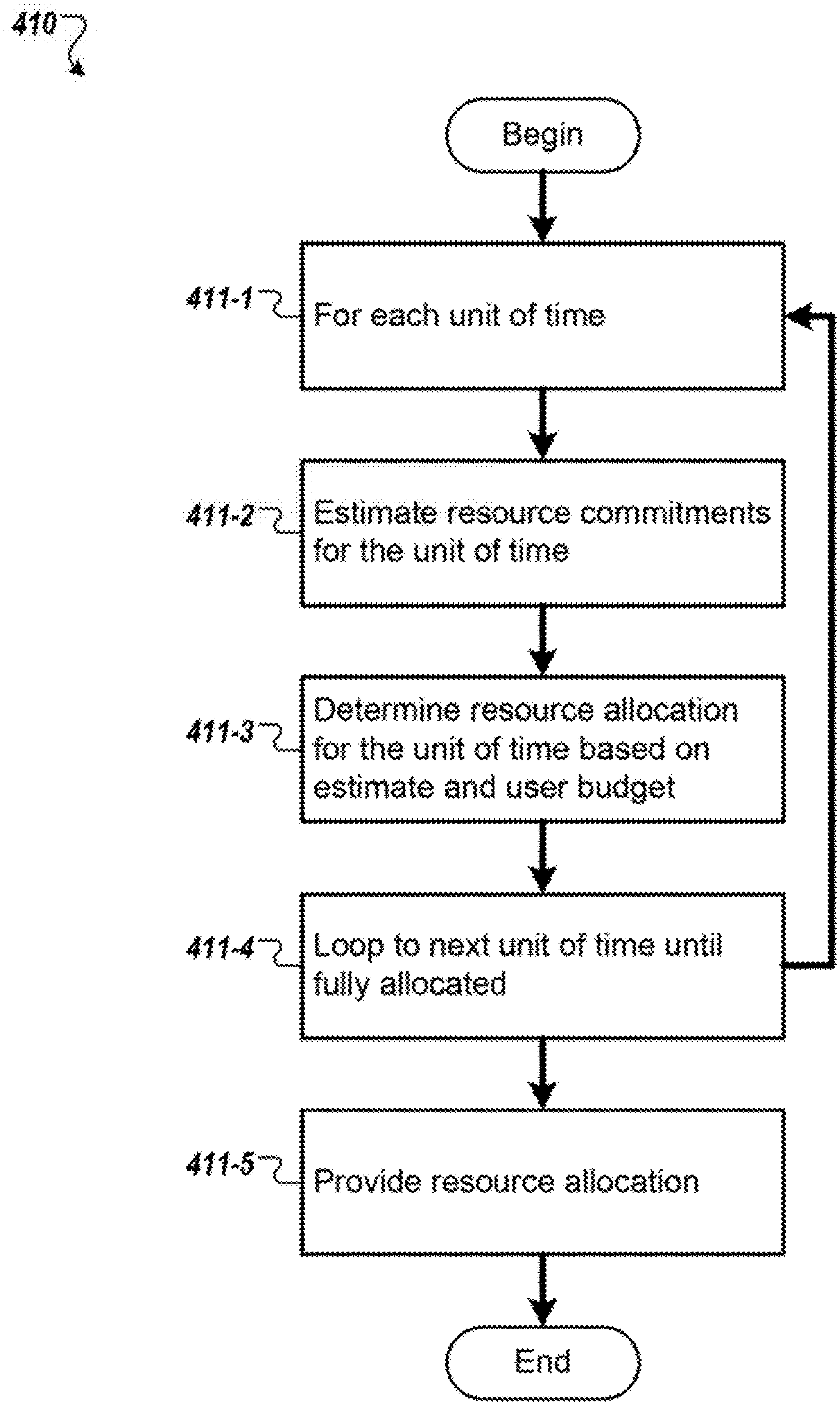

410
Begin
411-1
For each unit of time
411-2
Estimate resource commitments for the unit of time
411-3
Determine resource allocation for the unit of time based on estimate and user budget
411-4
Loop to next unit of time until fully allocated
411-5
Provide resource allocation
End

FIG. 4(c)

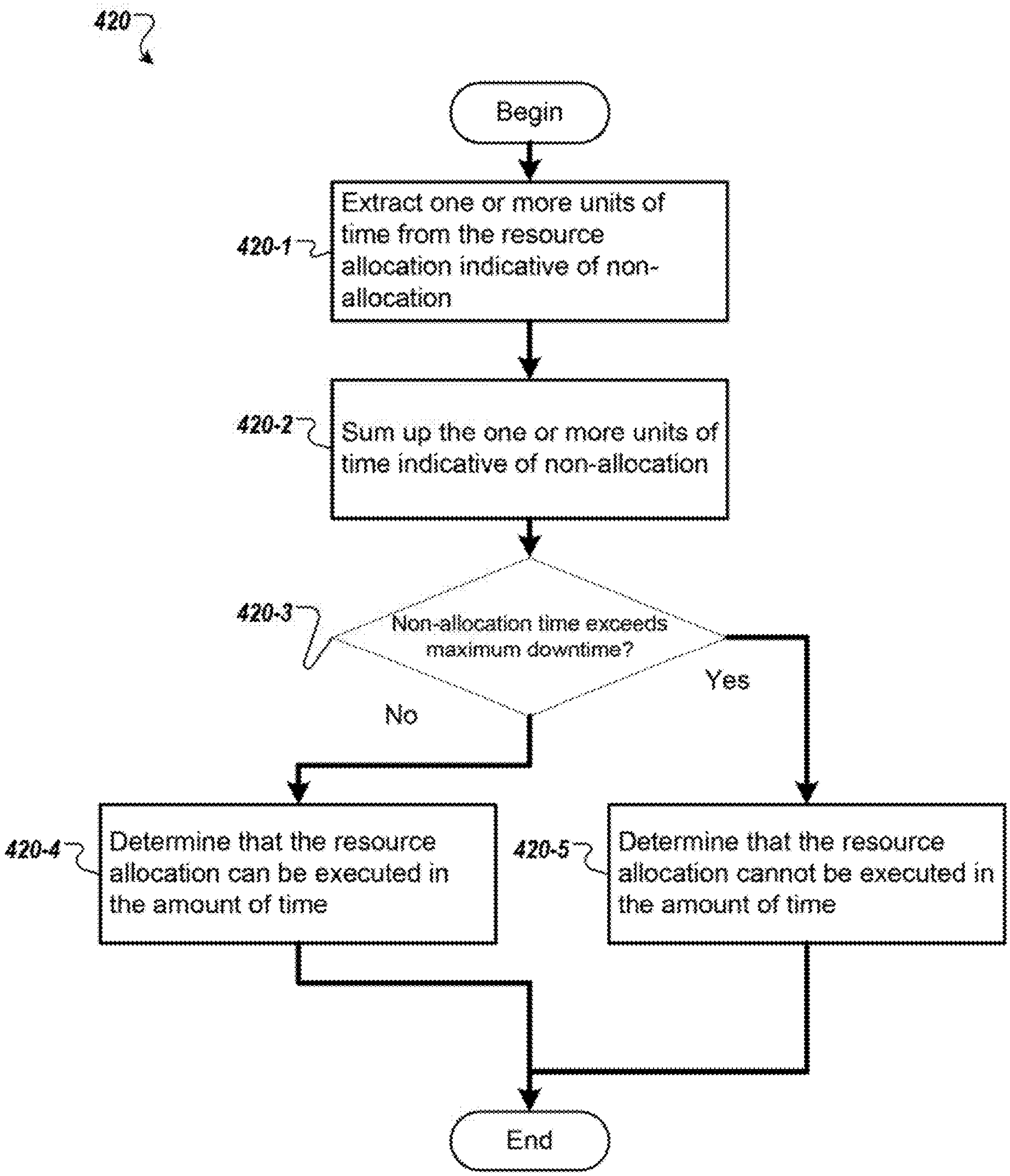

420
Begin
420-1
Extract one or more units of time from the resource allocation indicative of non-allocation
420-2
Sum up the one or more units of time indicative of non-allocation
420-3
Non-allocation time exceeds maximum downtime?
Yes
No
420-4
Determine that the resource allocation can be executed in the amount of time
420-5
Determine that the resource allocation cannot be executed in the amount of time
End

FIG. 4(d)

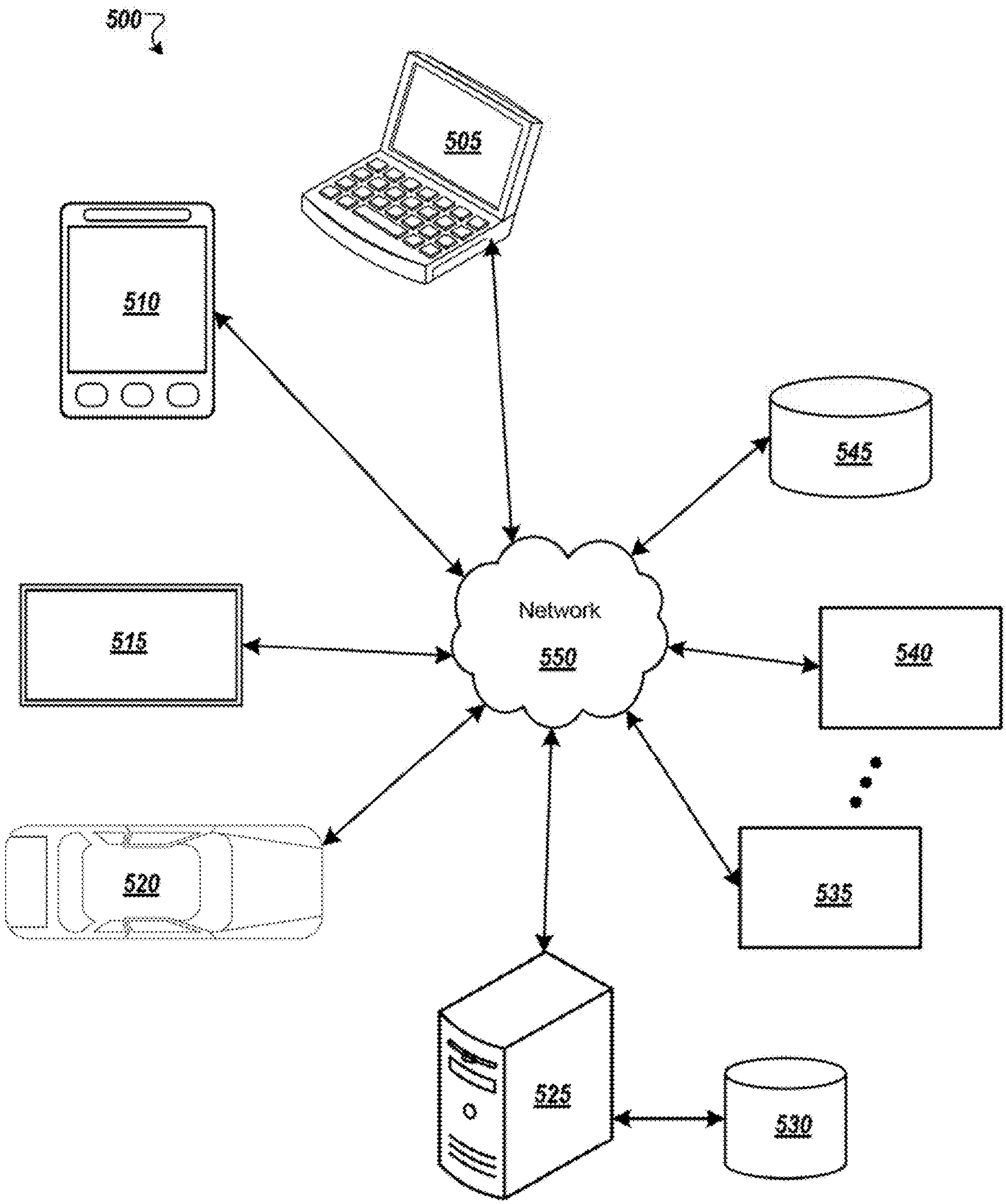
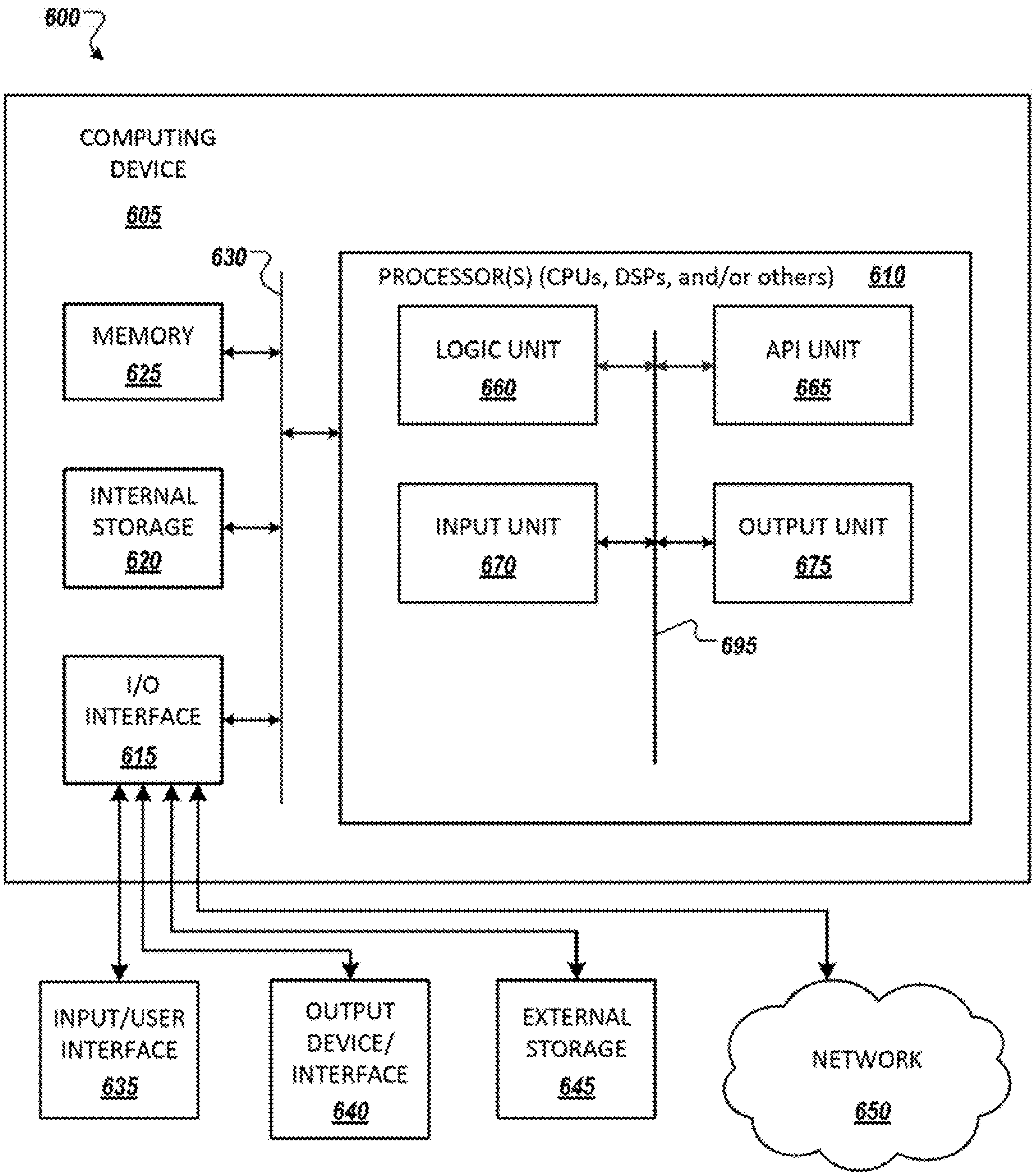


FIG. 5



600
COMPUTING DEVICE
605
630
PROCESSOR(S) (CPUs, DSPs, and/or others) 610
MEMORY
625
LOGIC UNIT
660
API UNIT
665
INTERNAL STORAGE
620
INPUT UNIT
670
OUTPUT UNIT
675
695
I/O INTERFACE
615
INPUT/USER INTERFACE
635
OUTPUT DEVICE/ INTERFACE
640
EXTERNAL STORAGE
645
NETWORK
650

FIG. 6

SYSTEMS AND METHODS FOR PROVISION OF A GUARANTEED BATCH

BACKGROUND

Field

The subject matter of the present disclosure relates generally to systems and methods for resource allocation and management, and more particularly, to system and methods for provision of a guaranteed batch.

Related Background

There are several related art implementations for scheduling optimizations for distributed computing systems. For example, related art implementations provide algorithms for optimizing scheduling to improve fairness, user satisfaction, and utilization. Related art implementations can also utilize deep learning to improve the packing of tasks with multiple resource requirements.

Related art budget based scheduling implementations can involve schedulers for workflows with deadline and/or budget guarantees using genetic algorithms. Other related art implementations may attempt to allocate resources based on the user Quality of Service (QoS) constraints, such as deadline and budget.

Related art implementations have also involved dynamic cloud implementations. In an example related art implementation, there is a service level agreement (SLA)-aware platform for resource management. There are related art implementations that involve a learning system that can make explicit Service Level Objectives (SLOs) based on historical data of periodic jobs and enforce such SLOs using scheduling algorithms.

SUMMARY

Example implementations described herein are directed to systems and methods to allocate resources for an intermittent load. The example implementations may involve a per user budget-based resource allocation that can obviate the need to provide deadline or load presence information on a per job basis.

The subject matter includes a computer-implemented method for managing a pool of resources, which can involve receiving a job request for execution on the pool of resources; determining an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources; determining a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time; determining that the job request is capable of being executed for the amount of time; and executing the job request over the amount of time, according to the resource allocation.

The subject matter further includes a non-transitory computer readable medium, storing instructions for managing a pool of resources, the instructions which can involve receiving a job request for execution on the pool of resources; determining an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources; determining a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time; determining that the job request is capable of being executed for the amount of time; and executing the job request over the amount of time, according to the resource allocation.

The subject matter further includes a system configured to manage a pool of resources, the system which can involve a processor, configured to receive a job request from the pool of resources; determine an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources; determine a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time; determine that the job request is capable of being executed for the amount of time; and execute the job request over the amount of time, according to the resource allocation.

The subject matter also includes a system for managing a pool of resources, the system which can involve means for receiving a job request for execution on the pool of resources; means for determining an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources; means for determining a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time; means for determining that the job request is capable of being executed for the amount of time; and means for executing the job request over the amount of time, according to the resource allocation.

The methods are implemented using one or more computing devices and/or systems. The methods may be stored in computer-readable media such as a non-transitory computer readable medium.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. **1**(*a*) and **1**(*b*) illustrate examples of resource multiplexing in accordance with an example implementation.

FIG. **2** illustrates an example cloud environment, in accordance with an example implementation.

FIGS. **3**(*a*) to **3**(*c*) illustrate example management information that can be utilized by the management system, in accordance with an example implementation.

FIGS. **4**(*a*) to **4**(*d*) illustrate examples of process implementations that can be executed by the management system, in accordance with an example implementation.

FIG. **5** shows an example environment suitable for some example implementations.

FIG. **6** shows an example computing environment with an example computing device suitable for use in some example implementations.

DETAILED DESCRIPTION

The subject matter described herein is taught by way of example implementations. Various details have been omitted for the sake of clarity and to avoid obscuring the subject matter. The examples shown below are directed to structures and functions for implementing systems and methods for provision of a guaranteed batch. Terms used throughout the description are provided as examples and are not intended to be limiting.

For example, the use of the term "automatic" may involve fully automatic or semi-automatic implementations involving user or administrator control over certain aspects of the implementation, depending on the desired implementation of one of ordinary skill in the art practicing implementations of the present application. Selection can be conducted by a user through a user interface or other input means, or can be implemented through a desired algorithm. Example implementations as described herein can be utilized either singularly or in combination and the functionality of the example implementations can be implemented through any means according to the desired implementations.

Cloud infrastructure workloads can often be broadly classified, in terms of their presence in the system, as either continuous or intermittent. Continuous workloads involve workloads that are always present in the system and can include both user-facing (e.g., latency-sensitive) server applications as well as continuous background (e.g., latency-tolerant) data-processing systems (e.g., data compression, web crawling/indexing).

Intermittent workloads are workloads that are characterized by varying periods of absence intercalated by periods of activity, where resources are actually required and/or consumed. This group mostly encompasses intermittent background data-processing systems (e.g., logs processing, simulations).

Load presence can be important for resource management (e.g., efficiency). Continuous workloads require continuous resource allocation, consuming a certain (usually varying over a determined interval) amount of resources for 24 hours/day. Variability in the consumption may lead to waste, as resources must be provisioned for the peak load. Cloud providers rely on different availability Service Level Objectives (SLO) to meet the user requirements without hurting overall fleet efficiency.

Intermittent workloads can be seen as a special class of continuous workloads with extreme consumption variability conditions (from 0 to peak). Therefore, continuous resource allocation can also be used for intermittent workloads, but such allocations can be very wasteful for intermittent workloads. Furthermore, the availability SLOs associated with continuous resource allocations may not be as meaningful for intermittent workloads as they are to continuous workloads, especially when the ratios between presence periods and absence periods approach zero.

Intermittent resource allocation, where resources are reserved for a given user for a pre-defined period and freed for the remaining time, is the solution to avoid increasing resource idleness time. However, intermittent resource allocation is much harder to achieve than continuous resource allocation, because it puts an extra burden on the users, requiring the users to specify not only the required amount of resources, but also when and for how long these resources will actually be needed, and on the provider, that must keep a complex scheduling of multiple intermittent resource allocations to be able decide on the feasibility of new user requests.

In related art implementations, large cloud infrastructures address intermittent load by using reclaimed resources. In these related art cases, the intermittently available resources are usually much cheaper than their continuous counterparts, in exchange for providing none or, at least, very weak SLOs.

Example implementations described herein are directed to a resource management model to allocate resources for intermittent load, or batch load, that relies on per user budgets and which may obviate the need for information regarding job duration or deadlines. Example implementations as described herein utilize a novel definition of a user budget, representing the maximum amount of resources a user can consume over a fixed period of time (e.g., constant 24 hours, but not limited thereto), hence, defined using a RESOURCE*TIME unit (e.g., CPU*HOURS, BYTES*HOUR, and so on).

Example implementations described herein can be applied to cloud providers and other entities (e.g., enterprises such as companies) with large distributed computing infrastructures attending multiple individual or groups of users to improve utilization/efficiency on the computing clusters.

FIGS. **1**(*a*) and **1**(*b*) illustrate an example of resource multiplexing in accordance with an example implementation. As illustrated in FIG. **1**(*a*), example implementations described herein time multiplex resources by transforming the shape of jobs to fit in the capacity of the system (e.g., spread the job request over the amount of time). Each job request may include information such as a list of tasks to be completed by the job request, a requested resource allocation, and one or more constraints on the job request. Each job may involve an aggregation of individual tasks (e.g., execute program A, conduct process B, feed result into process C, etc.), which may be run consecutively or concurrently, and which may be prioritized.

There is a high probability that at peak times, not all admitted jobs will fit into the capacity of the pool. To handle such situations, example implementations may disable individual tasks within jobs as needed to ensure that the SLOs of all admitted jobs are met, as shown in FIG. **3**(*c*) as discussed below. Disabled tasks can be marked on the user interface (UI) as "Disabled Due To Congestion," and such terminations will not count towards user configured failure limits for the job/task. In example implementations, jobs running in 'best effort' mode will be sized down or disabled completely, as necessary. If the system is still congested after such activity, jobs with guarantees which are farther away from their expected deadline (e.g., creation time+24 hours) will be disabled (e.g., squeezed) before the ones which are closer to make sure that no guarantees on job completion would be violated.

In the example of FIG. **1**(*b*), as users are assigned a guaranteed batch in terms of RESOURCE*TIME, the ceiling/commitment of the user is considered as an average throughput over a period of time (e.g., 24 hours). Accordingly, a commitment of Y resources means that the user may be able to consume, on average, Y units of this resource over the next period of time. The guaranteed batch workload may exceed Y when more resources are available and may be squeezed during contention. For example, if a user has a commitment of ten resources (e.g., Central Processing Units (CPUs), memory units, etc.) and is using all of the resources for a guaranteed batch, examples of the possible job consumption patterns are shown for FIG. **1**(*b*). Note that the resource usages over time (area of shaded area) are exactly the same in the three illustrated cases.

FIG. **2** illustrates an example cloud environment **200**, in accordance with an example implementation. In an example implementation, the cloud environment **200** can include one or more client devices **205**, a network **210**, one or more resource batch pools **215** (e.g., one or more servers) and a management system **220**. As used herein, "pools" may refer to one or more pools of resources, which may include (but are not limited to) one or more apparatuses or systems that may be configured to provide processing power with respect to one or more of CPUs, storage from storage systems in terms of flash memory or disk, specialized GPUs, specialized programs, or the like, depending on the desired implementation. The one or more client devices **205** execute one or more jobs on resources allocated from the one or more resource batch pools **215**, and may access such resources through network **210**. Examples of such client devices **205** are provided with respect to FIG. **5**. Network **210** may be configured to facilitate connections between the one or more client devices **205**, the one or more resource batch pools **215** and the management system **220**. Such a network **210** may be in the form of a Local Area Network (LAN), a Wide Area Network (WAN), or otherwise depending on the desired implementation. The one or more resource batch pools **215** may be implemented in the form of a cloud including any combination of dedicated central processing units (CPUs), storage systems, memory units, general processing units (GPUs), servers, and other hardware units that are configured to provide desired resources for the one or more client devices **205**. Resources that can be provided can include, but are not limited to, CPUs, memory, bandwidth, specialized processors, and so on according to the desired implementation.

In example implementations, the one or more resource batch pools **215** may be managed by a management system **220**. Management system **220** can be configured to receive requests for a resource allocation for a job from the one or more client devices **205**, and then manage the allocation of resources from the resource batch pools **215** for the one or more clients **205** based on the example implementations described herein, and as shown in FIG. **3**(*c*) and further discussed below. For example, a resource allocation may provide an indication of resources that are required to execute the job associated with the job request, and may be specified in terms of RESOURCE*TIME (e.g., CPU*TIME, MEMORY*TIME, etc.), and/or may be specified in terms of resources (e.g., CPU, MEMORY).

In example implementations, a batch pool is utilized as an entity holding the resources available for users with batch load as illustrated by the resource batch pool **215** of FIG. **2**, and provides a pool of resources for use. The capacity from a given batch pool is obtained by blocking computing resources for a specified amount of time Δt, hence defined as RESOURCE*HOUR/Δt. Such a batch pool provides different products for the pool of resources (e.g., CPU, memory, disk, etc.). For example, the pool of resources may include one or more apparatuses or systems configured to provide any combination of processing power in terms of CPUs, storage from storage systems in terms of flash memory or disk, specialized GPUs, specialized programs, and so on depending on the desired implementation. The capacity of each batch product p, bc_p, expressed in RESOURCE*HOUR/period of time Δt units, is defined in terms of the capacity allocated from the underlying continuous product for the period Δt, cc_p, as:

$$bc_p=(cc_p*\Delta t) \tag{1}$$

For example, a pool administrator can fund a batch pool by allocating 1000 RESOURCE units of a continuous product p, over a period of time (e.g., 24 hours). The resulting capacity of the equivalent batch product would be:

bc_p=(1000*24) or 24000 RESOURCE*HOURs/DAY

The batch pool capacity can be defined as the integral of the instantaneous capacity allocated from the underlying continuous product p at any moment t, $cc_p(t)$, over a period of time Δt:

$$bc_p = \int_{t-\Delta t}^{t} cc_p(t)dt \tag{2}$$

Using this definition the 24000 RESOURCE*HOURs/DAY, batch capacity mentioned in the previous example could be obtained both by allocating 1000 RESOURCE units for 24 hours or 2000 RESOURCE units for 12 hours over a period of 24 hours, for example.

Similarly, the fraction of the batch pool capacity allocated for a particular user, the user commitments, over a period of time Δt can be expressed as:

$$bc_{up} = \int_{t-\Delta t}^{t} cc_{up}(t)dt \tag{3}$$

In the example implementations, users initially start with the same commitments (e.g., 0) and have the number adjusted periodically according to their past consumption. Such a definition does not imply any particular bound on the instantaneous capacity of the underlying continuous product at any moment t.

Batch Pool capacity (bc_p) and Batch User Commitments (bc_{up}) are concepts valid over a period of time Δt, and no assumptions are made about the underlying instantaneous values at any particular instant t.

In example implementations, the proposed resource allocation mode can provide two SLOs for users with intermittent load. In a first SLO, users with sufficient load will be able to consume their entire budget (in this case, commitments) over a period Δt. In a second SLO, the downtime for any job j, dj, over a period of time Δt does not exceed the maximum specified downtime D. A job can be considered as experiencing downtime if a particular task in the job request cannot access the minimum required resources to execute the task, when overall resource allocation for the job falls below a certain threshold, or can be defined by other methods depending on the desired implementation.

The submitted load, sl, is defined as all of the load a user submits to a batch pool and admitted load, al, is defined as the amount of the sl admitted by the batch scheduler to run under an SLO. In an example implementation and as shown in FIG. **3**(*c*) and discussed below, users can submit jobs that specify the required resources to start running the job and the maximum downtime D the job can take. In such an example, the first SLO can be utilized by enforcing the following two inequalities during resource allocation:

$$\forall\, t, \int_{t-\Delta t}^{t} al_{pu}(t)dt \geq \text{MIN}\left(bc_{pu}, \int_{t-\Delta t}^{t} sl_{pu}(t)dt\right) \tag{4}$$

$$\forall\, t,\ al_{pu}(t) \leq sl_{pu}(t) \tag{5}$$

Inequality (4) specifies that for every moment t, the admitted load for user u over the last Δt period of time is the minimum of the commitments and the submitted load. That is, if (submitted load≤commitments), all submitted load is admitted by the system. On the contrary, if commitments<submitted load, only the load amounting to the commitments will be accepted to receive the SLO.

Inequality (5) is a control mechanism to prevent allocating more resources than the user can actually consume at any moment t. The second SLO imposes a constraint on how long jobs can be delayed and can be obtained with job prioritization according to the remaining user budget over ∀t.

Based on the equations above, the criteria C(u) to identify the user u to have the next job released is:

$$C = \mathrm{MAX}\left(0,\ bc_{pu} - \int_{t-\Delta t}^{t} cc_{pu}(t)dt\right)$$

The user with the greatest C(u) is ahead of the queue. For a given user, the jobs are dequeued in first in first out (FIFO) order.

FIG. **3**(*a*) illustrates example management information as utilized by the management system, in accordance with an example implementation. Specifically, FIG. **3**(*a*) illustrates example management information for management of resource allocation for one or more users. The management information of FIG. **3**(*a*) facilitates the management system **220** to manage the resource budget for each user of a plurality users of the pool of resources through managing resource allocation to one or more client devices associated with each user based on the resource budget associated with each user. As the client devices may involve multiple different users, or as a user may utilize multiple client devices to execute various jobs, resource budgets can be managed at a user level as described below.

The management information as illustrated in FIG. **3**(*a*) can include user identifiers (IDs) **300**, an associated resource allocation budget **301**, present resource commitments **302**, historical resource usage **303**, and pending job requests **304** for resource allocation. The user ID **300** can indicate the identification of the user that has job requests running on the resource batch pools **215**. The resource allocation budget **301** can be represented as a vector across all of the different types of resources available in the resource batch pools **215**, with each element in the vector indicative of the amount of available resources for the particular type of resource. Such resources can be represented as RESOURCE*TIME, with the time based on the time period associated with the type of resource, and calculated based on equation (3).

The present resource commitments can include the jobs presently executed by the user and the resources consumed, as illustrated in FIG. **1**(*b*). The present resource commitments **302** may also include a vector across all of the different types of resources available in the resource batch pools **215**, with each element in the vector indicative of the amount of resources consumed for the particular type of resource.

Such values can be determined from the summation of the areas of the resources utilized over time for each type of resource for each job as illustrated in FIG. **1**(*b*). Historical resource usage **303** can include the jobs historically executed by the user and the associated historical resource usage or consumption as illustrated in FIG. **1**(*b*). Once jobs are completed for the user, the job can be moved from the present resource commitments **302** to the historical resource usage **303**.

Job requests **304** can include pending job requests for resource allocation submitted by the client device associated with the user ID. The job requests **304** may eventually be processed in accordance with the flow of FIG. **4**(*a*), as explained below.

FIG. **3**(*b*) illustrates example management information for management of the pool of resources for the one or more resource batch pools **215**, in accordance with an example implementation. Such management information can include types of resources **310**, time period for each type of resource **311**, current resource commitments for the type of resource **312**, historical resource usage **313**, and expected resource commitments **314**. Types of resources **310** can include the type of resource available for resource batch pools **215**, such as storage and central processing units (CPUs) provided from a cloud pool, GPUs, bandwidth, or other resources depending on the desired implementation, as well as the total resources for the type of resource in the resource batch pool **215** (e.g., the batch pool capacity for the type of resource calculated from equation (2)). The time period for each type of resource **311** can include the set time period (e.g., 24 hours) for each of the resources, and the unit of time associated with the resource type (e.g., 1 hour), to define the resource availability in terms of RESOURCE*TIME. The time period and unit of time can be set by an administrator or though other methods in accordance with the desired implementation.

Current resource commitments **312** can include the job information as illustrated in FIG. **1**(*b*) across all users of the resource batch pool **215** that utilize the particular type of resource. Current resource commitments **312** can also include a value indicating how much of the particular type of resource is consumed in terms of RESOURCE*TIME.

Historical resource usage **313** can include the job information as illustrated in FIG. **1**(*b*) from all of the jobs that utilize the particular type of resource. When a job in the current resources commitments is completed, such job information can be moved to the historical resource usage **313**.

Expected resource commitments **314** can include information as illustrated in FIG. **1**(*b*) that is calculated based on historical resource usage **313**. Such a calculation can be conducted by taking an average of resources consumed for each time unit for a time period, or through other methods, depending on the desired implementation.

FIG. **3**(*c*) illustrates an example breakdown of a job request, in accordance with an example implementation. Each job request submitted by a client device is associated with a job request ID **304-1** and can include information provided by the client device such as a list of tasks **304-2** to be completed by the job request, a requested resource allocation **304-3**, and one or more constraints on the job request **304-4**. Each job may involve an aggregation of individual tasks **304-2** (e.g., execute program A, conduct process B, feed result into process C, etc.), which can run (e.g., at one or more servers) consecutively or concurrently in any combination, depending on the specifications of the job request. Tasks may also be prioritized in terms of importance, so that users can prioritize which tasks are to be disabled last, which tasks cannot be disabled for the job request, and so on, depending on the desired implementation.

Resource allocation **304-3** can indicate the required resources to execute the job associated with the job request ID. Resource allocation **304-3** can be specified in terms of RESOURCE*TIME as described above (e.g., CPU*TIME, MEMORY*TIME), and/or can be specified in terms of resources (e.g., CPU, MEMORY) depending on the desired implementation. For example, but not by way of limitation, the required resources may be provided at a server, in response to the information provided by the client device.

Constraints **304-4** can indicate constraints associated with the job request ID, such as the maximum downtime the job or particular tasks are allowed to have, time constraints in executing the job request, specified SLOs and resource minimums for tasks, and so on, depending on the desired implementation.

FIGS. **4**(*a*) to **4**(*d*) illustrate examples of process implementations that can be executed by the management system, in accordance with an example implementation.

FIG. **4**(*a*) illustrates an example of a process implementation **400**, in accordance with an example implementation. The process **400** of FIG. **4** illustrates an example for a process for the flow at FIG. **2**. At **405**, the process receives a job request for execution on the pool of resources. Such a request may be received from the one or more clients **205** by the management system **220**. Such a job request can involve a list of tasks, constraints for the job request, and a request of at least one of processing, memory, bandwidth, GPUs or specialized processors, and so on, depending on the desired implementation, and received by the management system **220** to be managed as illustrated in FIG. **3**(*c*).

At **410**, the process determines the amount of time utilized for execution of the job request. The process can refer to the available resources from the pool of resources and historical usage data of the pool of resources from FIG. **3**(*b*). In an example, the available units of resource time can be determined for the resource batch pool **215** based on the available resources from the pool of resources (e.g., total resource units associated with a type of resource from FIG. **3**(*b*)) and the estimated resource commitments over one or more units of time based on historical resource usage **313** of the pool of resources as indicated in estimated resource commitments of FIG. **3**(*b*).

Moreover, if a constraint in the job request requires that the job be executed within a certain period of time, the time provided in the constraint can be utilized as the amount of time utilized for execution. Depending on the desired implementation, time may also be a constant (e.g., 24 hours) in accordance to how RESOURCE*TIME is determined in the desired implementation.

The available units of resource time can be determined from the difference between the total resource units the estimated resource commitments. Similarly, available units of resource time for a user ID associated with the job request can be determined based on a resource allocation budget assigned to the user associated with the job request and resource commitments of the user associated with the job request as indicated in FIG. **3**(*a*). The available units of resource time can be determined from the difference between the budget and the resource commitments.

At **415**, the process determines resource allocation from the pool of resources based on the amount of time utilized for the job request. Depending on the present resource commitments of the user associated with the request and the expected resource commitments **314** over the time period, an allocation is determined as illustrated in FIGS. **1**(*a*) and **1**(*b*) that spreads the job request over the period of time.

As illustrated in FIGS. **1**(*a*) and **1**(*b*), the job request does not have to be fulfilled in uniform allocations; any process can be utilized to allocate any amount of resources for a particular period of time to spread the job request over the period of time. For example, the job request can be spread in a greedy manner, wherein the job request is given all of the expected free resources for a particular unit of time based on the historical usage, and constrained by the RESOURCE*TIME budget of the associated user or up to the resources needed to execute the job. The job request may also be spread according to priority of other tasks determined to be executed in that particular unit of time. Other operations for spreading the job request may be substituted therefor, as would be understood by those skilled in the art.

As illustrated in FIG. **4**(*c*), the determining the resource allocation can involve a process that for each unit of time, determines the resource allocation for the each unit of time based on a budget associated with the job request and available resources from the pool of resources for the each period of time. As illustrated in FIG. **4**(*b*), the determining the resource allocation from the pool of resources can also involve, for each unit of time, determining the resource allocation for the job request based on the estimated available resources for the each unit of time.

Depending on the desired implementation, queuing is also applied for the resource allocation. The resource allocation from the pool of resources is determined based on the queuing of the user from the plurality of the users associated with the job request, and the other expected commitments received from other users, based on the proposed criteria C(u). In conjunction with the example implementations described in FIGS. **4**(*b*) and **4**(*c*), the resource allocation can be conducted for each unit of time while taking queuing into account. For example, for each unit of time, the process can estimate the queuing of the user associated with the job request based on the proposed criteria C(u); and estimate resource commitments of users from the plurality of users ahead of the user associated with the job request in the queue.

At **420**, the process determines whether the job request is capable of being executed for the amount of time. That is, a check is performed to determine if the proposed amount of time to execute the job request meets with the SLO requirement of the job request or the user as defined by the constraints **304-4** from FIG. **3**(*c*), and/or if the resource request is valid (e.g., can be executed over the specified time period).

For example, but not by way of limitation, it may not be possible to execute the job request within the time defined by the constraint associated with a job request, whereupon the job may be returned with a message indicative of the constraint not being possible. Such a determination can be made by comparing the available RESOURCE*TIME units of the user with the requested RESOURCE*TIME units, and also by comparing the RESOURCE*TIME units expected to be free during the time period defined by the constraint with the requested RESOURCE*TIME units. If there is insufficient RESOURCE*TIME units in the pool of resources, or the user does not have the sufficient RESOURCE*TIME units for the given time constraint, then the job is considered to be invalid.

It may also not be possible to execute the job request within the time defined by the administrator of the management system as the basis for determining RESOURCE*TIME (e.g., 24 hours), in which case the job may be returned with a message indicative of the job not being executable at that time, or that the user/pool of resources does not have sufficient resources to execute the job request.

At **425**, should the process determine that the job request is capable of being executed for the amount of time, the process executes the job request over the amount of time according to the resource allocation. The executing the job request can involve, for each unit of time in the amount of time, allocating resources from the pool of resources according to the determined resource allocation corresponding to the each unit of time, as illustrated in FIG. **1**(*a*).

Other executions of the job request may also be possible, depending on the desired implementation. For example, if the predicted usage is more than the currently experienced resource consumption, more resources may be allocated to the job request for a unit of time according to the desired implementation (e.g., proportionally to the other job requests being executed, based on priority of job requests currently executed, etc.).

If the predicted usage is less than the currently experienced resource consumption, then job requests may be given proportionately less resources based on priority, or by other metrics according to the desired implementation. In such an example implementation, the execution of the job request at **425** can involve determining if the currently experienced resource consumption is less than the predicted resource consumption derived from historical usage. If so, then job requests executed during the unit of time can be allocated more resources according to the desired implementation. If not, then job requests executed during the unit of time can be provided fewer resources in a manner according to the desired implementation.

Depending on the desired implementation, queuing may be employed for the received plurality of requests. In such an example implementation, the process manages the resource budget for each of the users as illustrated in FIG. **3**(*a*), and determines the queuing order for the job requests based on the user associated with the job request. The queuing is determined based on resource commitments of the plurality of users and the resource budget for the each user of the plurality of users as described with respect to the proposed criteria C(u).

FIG. **4**(*b*) illustrates a process for determining the amount of time to be utilized for executing the job request, in accordance with an example implementation. The process begins by, for each unit of time, determining the available units of resource time at **410-1**. At **410-2**, the process then determines available resources for the job request from the pool of resources based on available units of resource time from referring to management information of FIGS. **3**(*a*) and **3**(*b*). Through reference to the management information as illustrated in FIGS. **3**(*a*) and **3**(*b*), the available resources from the pool of resources can be determined and the process can estimate resource commitments over one or more units of time based on historical resource usage **313** of the pool of resources. For example, the resource commitments of the user can be estimated based on the historical resource usage **313** of the resource batch pool **215** as described with respect to the expected resource commitments **314** of FIG. **3**(*b*), and the process can estimate the available resources from the pool of resources from the estimated resource commitments, which can involve taking the difference between the total resources available and the estimated resource commitments.

At **410-3**, the process then determines a period of time that is capable of being allocated for the job request based on a budget of resources allocated for the job request and the available units of resource time.

FIG. **4**(*c*) illustrates an example flow for determining the period of time for a job request and conducting the resource allocation at **415**. Specifically, FIG. **4**(*c*) illustrates an example process for the flow of **410** and **415** for FIG. **4**(*c*), as an example flow for determining the period of time that is capable of being allocated for the job request based on a budget of resources allocated for the job request and the available units of resource time comprises. At **411-1**, a loop is initiated for each unit of time, the unit of time defined by the time period set in the management information of FIG. **3**(*b*). At **411-2**, the process estimates the resource commitments for the unit of time. The estimation can be referenced from the estimation of resource commitments from the management information of FIG. **3**(*b*).

At **411-3**, a resource allocation is determined for the unit of time. Such a resource allocation can be determined based on the resource commitments of the user associated with the job request for the each unit of time and a resource allocation budget assigned to the user associated with the job request for the each unit of time from referring to the present commitments of the user ID associated with the job request, the budget for the user ID, the total resources for the type of resource in the resource batch pool **215**, and the current and expected commitments for that type of resource from the management information of FIGS. **3**(*a*) and **3**(*b*).

The current and expected commitments for that type of resource are subtracted from the total resources to determine the expected available resources for the unit of time. Similarly, the present user commitments are subtracted from the user budget to determine the expected resources available to the user for that time unit. If the user can consume the resource budget (e.g., expected resources available to the user<expected available resources for the unit of time), then resources up to the expected resources available to the user are allocated as indicated in equations (4) and (5). Otherwise, the allocation is done based on the SLO policy as indicated in equation (5).

At **411-5**, after all of the allocation is completed, the amount of time units utilized is determined from the resource allocation as the period of time to be associated with the job request, and the proposed resource allocation is provided.

FIG. **4**(*d*) illustrates example process for evaluating whether the determined period of time is acceptable. Such a process can be utilized at the process of **420**. In an example as to whether the determined period of time is acceptable, one method is to determine whether a downtime associated with the resource allocation does not exceed the maximum downtime associated with the job request.

At **420-1**, the process extracts one or more units of time from the resource allocation indicative of non-allocation (e.g., when zero resources are allocated for that particular time unit). At **420-2**, the process sums up the one or more units of time that are indicative of non-allocation to determine the estimated downtime. At **420-3**, a determination is made as to whether the non-allocation time (estimated downtime) exceeds the maximum downtime. For the downtime not exceeding the maximum downtime (No), the process proceeds to **420-4** and determines that the resource allocation can be executed in the amount of time, therefore capable of being executed. Otherwise, for the downtime exceeding the maximum downtime (Yes), the flow proceeds to **420-5** and determines that the job request is not capable of being executed.

Other methods can be utilized in conjunction with, or separately from FIG. **4**(*d*) to determine if the proposed period of time and resource allocation is acceptable. Such a determination can be made based on whether or not the SLOs as described above are met. The job request may be associated with a time requirement, so the determination can be based on whether the amount of time to be utilized for executing the job request meets a time requirement associated with the job request. Another example for a determination whether the job request is capable of being executed for the amount of time can be based on whether the proposed resource allocation can be executed (e.g., is invalid or not due to violating constraints associated with present job commitments, other users, and so on).

In some examples, process **400** may be implemented with different, fewer, or more blocks. Process **400** may be implemented as computer executable instructions, which can be stored on a medium, loaded onto one or more processors of one or more computing devices, and executed as a computer-implemented method.

FIG. **5** shows an example environment suitable for some example implementations. Environment **500** includes devices **505-545**, and each is communicatively connected to at least one other device via, for example, network **550** (e.g., by wired and/or wireless connections). Some devices may be communicatively connected to one or more storage devices **530** and **545**. Such devices **505-545** can be examples of client devices **205**, or can also be a device utilized to facilitate the functionality of the management system **220**.

An example of one or more devices **505-545** may be computing device **605** described below in FIG. **6**. Devices **505-545** may include, but are not limited to, a computer **505** (e.g., a laptop computing device), a mobile device **510** (e.g., smartphone or tablet), a television **515**, a device associated with a vehicle **520**, a server computer **525**, computing devices **535-540**, storage devices **530** and **545**.

FIG. **6** shows an example computing environment with an example computing device suitable for use in some example implementations, such as management system **220** or client devices **205**. Computing device **605** in computing environment **600** can include one or more processing units, cores, or processors **610**, memory **615** (e.g., RAM, ROM, and/or the like), internal storage **620** (e.g., magnetic, optical, solid state storage, and/or organic), and/or I/O interface **625**, any of which can be coupled on a communication mechanism or bus **630** for communicating information or embedded in the computing device **605**.

Computing device **605** can be communicatively coupled to input/user interface **635** and output device/interface **640**. Either one or both of input/user interface **635** and output device/interface **640** can be a wired or wireless interface and can be detachable. Input/user interface **635** may include any device, component, sensor, or interface, physical or virtual, that can be used to provide input (e.g., buttons, touch-screen interface, keyboard, a pointing/cursor control, microphone, camera, braille, motion sensor, optical reader, and/or the like). Output device/interface **640** may include a display, television, monitor, printer, speaker, braille, or the like. In some example implementations, input/user interface **635** and output device/interface **640** can be embedded with or physically coupled to the computing device **605**. In other example implementations, other computing devices may function as or provide the functions of input/user interface **635** and output device/interface **640** for a computing device **605**.

Examples of computing device **605** may include, but are not limited to, highly mobile devices (e.g., smartphones, devices in vehicles and other machines, devices carried by humans and animals, and the like), mobile devices (e.g., tablets, notebooks, laptops, personal computers, portable televisions, radios, and the like), and devices not designed for mobility (e.g., desktop computers, other computers, information kiosks, televisions with one or more processors embedded therein and/or coupled thereto, radios, and the like). [96] Computing device **605** can be communicatively coupled (e.g., via I/O interface **625**) to external storage **645** and network **650** for communicating with any number of networked components, devices, and systems, including one or more computing devices of the same or different configuration. Computing device **605** or any connected computing device can be functioning as, providing services of, or referred to as a server, client, thin server, general machine, special-purpose machine, or another label.

The I/O interface **625** may include wireless communication components (not shown) that facilitate wireless communication over a voice and/or over a data network. The wireless communication components may include an antenna system with one or more antennae, a radio system, a baseband system, or any combination thereof. Radio frequency (RF) signals may be transmitted and received over the air by the antenna system under the management of the radio system.

I/O interface **625** can include, but is not limited to, wired and/or wireless interfaces using any communication or I/O protocols or standards (e.g., Ethernet, 802.11x, Universal System Bus, WiMax, modem, a cellular network protocol, and the like) for communicating information to and/or from at least all the connected components, devices, and network in computing environment **600**. Network **650** can be any network or combination of networks (e.g., the Internet, local area network, wide area network, a telephonic network, a cellular network, satellite network, and the like).

Computing device **605** can use and/or communicate using computer-usable or computer-readable media, including transitory media and non-transitory media. Transitory media include transmission media (e.g., metal cables, fiber optics), signals, carrier waves, and the like. Non-transitory media include magnetic media (e.g., disks and tapes), optical media (e.g., CD ROM, digital video disks, Blu-ray disks), solid state media (e.g., RAM, ROM, flash memory, solid-state storage), and other non-volatile storage or memory.

Computing device **605** can be used to implement techniques, methods, applications, processes, or computer-executable instructions in some example computing environments. Computer-executable instructions can be retrieved from transitory media, and stored on and retrieved from non-transitory media. The executable instructions can originate from one or more of any programming, scripting, and machine languages (e.g., C, C++, C#, Java, Visual Basic, Python, Perl, JavaScript, and others).

Processor(s) **610** can execute under any operating system (OS) (not shown), in a native or virtual environment. One or more applications can be deployed that include logic unit **660**, application programming interface (API) unit **665**, input unit **670**, output unit **675**, and inter-unit communication mechanism **695** for the different units to communicate with each other, with the OS, and with other applications (not shown). The described units and elements can be varied in design, function, configuration, or implementation and are not limited to the descriptions provided.

In an example implementation of a management system **220**, external storage **645** may be configured to store the management information as illustrated in FIGS. **3**(*a*) and **3**(*b*), and processor(s) **610** may be configured to execute the functions as depicted in FIGS. **4**(*a*) to **4**(*d*).

In some example implementations, when information or an execution instruction is received by API unit **665**, it may be communicated to one or more other units (e.g., logic unit **660**, input unit **670**, and output unit **675**.

In some instances, logic unit **660** may be configured to control the information flow among the units and direct the services provided by API unit **665**, input unit **670**, and output unit **675**, in some example implementations described above. For example, the flow of one or more processes or implementations may be controlled by logic unit **660** alone or in conjunction with API unit **665**.

Any of the software components described herein may take a variety of forms. For example, a component may be a stand-alone software package, or it may be a software package incorporated as a "tool" in a larger software product. It may be downloadable from a network, for example, a website, as a stand-alone product or as an add-in package for installation in an existing software application. It may also be available as a client-server software application, as a web-enabled software application, and/or as a mobile application.

Although a few example implementations have been shown and described, these example implementations are provided to convey the subject matter described herein to people who are familiar with this field. It should be understood that the subject matter described herein may be implemented in various forms without being limited to the described example implementations. For example, but not by way of limitation, the foregoing equations, expressions, and numerical examples are for illustrative purposes, and others may be substituted therefor, as would be understood by those skilled in the art.

The subject matter described herein can be practiced without those specifically defined or described matters or with other or different elements or matters not described. It will be appreciated by those familiar with this field that changes may be made in these example implementations without departing from the subject matter described herein as defined in the appended claims and their equivalents.

What is claimed is:

1. A computer-implemented method for managing a pool of resources, the method comprising:

receiving, at data processing hardware in communication with the pool of resources via a network, a job request for execution on the pool of resources, the job request received from a client device in communication with the data processing hardware and the pool of resources via the network, the pool of resources comprising computing resources and/or storage resources;

determining, by the data processing hardware, an amount of time to be utilized for executing the job request by, for each unit of time:

determining available units of resource time based on available resources from the pool of resources and estimated resource commitments over one or more units of time based on historical resource usage of the pool of resources;

determining available resources for the job request from the pool of resources based on the available units of resource time; and

determining a period of time that is capable of being allocated for the job request based on a budget of resources allocated for the job request and the available units of resource time;

determining, by the data processing hardware, a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time;

determining, by the data processing hardware, that the job request is capable of being executed for the amount of time; and

allocating, by the data processing hardware, resources from the pool of resources to the client device according to the resource allocation, the client device configured to execute the job request on the allocated resources over the amount of time,

wherein the determining the available units of resource time comprises:

determining a first available units of resource time based on available resources from the pool of resources and the estimated resource commitments over one or more units of time based on the historical resource usage of the pool of resources; and

determining a second available units of resource time based on available units of resource time of a user associated with the job request, the available units of resource time of the user associated with the job request determined based on a resource allocation budget assigned to the user associated with the job request and resource commitments of the user associated with the job request.

2. The computer-implemented method of claim **1**, wherein the determination that the job request is capable of being executed for the amount of time is based on a determination that the amount of time to be utilized for executing the job request meets a time requirement associated with the job request.

3. The computer-implemented method of claim **1**, wherein the determination that the job request is capable of being executed for the amount of time is based on a determination that a downtime associated with the resource allocation does not exceed a maximum downtime associated with the job request.

4. The computer-implemented method of claim **3**, wherein the determination that a downtime associated with the resource allocation does not exceed the maximum downtime associated with the job request comprises:

determining the downtime of the resource allocation based on determining one or more units of time in the resource allocation being indicative of non-allocation;

for the downtime not exceeding the maximum downtime, determining that the job request is capable of being executed; and

for the downtime exceeding the maximum downtime, determining that the job request is not capable of being executed.

5. The computer-implemented method of claim **1**, wherein the determination that the job request is capable of being executed for the amount of time is based on a determination that the resource allocation can be executed.

6. The computer-implemented method of claim **1**, wherein determining the resource allocation comprises, for each unit of time, determining the resource allocation for the each unit of time based on a budget associated with the job request and available resources from the pool of resources for the each period of time.

7. The computer-implemented method of claim **1**, wherein allocating the resources from the pool of resources to the client device comprises, for each unit of time in the amount of time, allocating the resources from the pool of resources according to the resource allocation corresponding to the each unit of time.

8. The computer-implemented method of claim **1**, further comprising:

managing, by the data processing hardware, a resource budget for each user of a plurality users of the pool of resources; and

determining, by the data processing hardware, queuing for the plurality of users based on resource commitments of the plurality of users and the resource budget for the each user of the plurality of users.

9. The computer-implemented method of claim **8**, wherein the job request comprises a request of at least one of processing and memory.

10. The computer-implemented method of claim **8**, wherein managing the resource budget for each user of a plurality users of the pool of resources comprises managing resource allocation to one or more client devices associated with each user based on the resource budget associated with each user.

11. The computer-implemented method of claim **8**, wherein determining the resource allocation from the pool of resources is based on the queuing of the user from the plurality of the users associated with the job request.

12. A computer-implemented method for managing a pool of resources, the method comprising:

receiving, at data processing hardware in communication with the pool of resources via a network, a job request for execution on the pool of resources, the job request received from a client device in communication with the data processing hardware and the pool of resources via the network, the pool of resources comprising computing resources and/or storage resources;

determining, by the data processing hardware, an amount of time to be utilized for executing the job request by, for each unit of time:

determining available units of resource time based on available resources from the pool of resources and estimated resource commitments over one or more units of time based on historical resource usage of the pool of resources;

determining available resources for the job request from the pool of resources based on the available units of resource time; and

determining a period of time that is capable of being allocated for the job request based on a budget of resources allocated for the job request and the available units of resource time by:

for each unit of time, determining the resource allocation from the pool of resources based on the available resources for the each unit of time and the estimated commitments for the each unit of time until the job request is fully allocated; and

determining the period of time from the resource allocation;

determining, by the data processing hardware, a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time;

determining, by the data processing hardware, that the job request is capable of being executed for the amount of time; and

allocating, by the data processing hardware, resources from the pool of resources to the client device according to the resource allocation, the client device configured to execute the job request on the allocated resources over the amount of time,

wherein for each unit of time, the determining the resource allocation from the pool of resources based on the available resources for the each unit of time and the estimated commitments for the each unit of time until the job request is fully allocated further comprises for the each unit of time, determining the resource allocation based on the resource commitments of the user associated with the job request for the each unit of time and a resource allocation budget assigned to the user associated with the job request for the each unit of time.

13. A computer-implemented method for managing a pool of resources, the method comprising:

receiving, at data processing hardware in communication with the pool of resources via a network, a job request for execution on the pool of resources, the job request received from a client device in communication with the data processing hardware and the pool of resources via the network, the pool of resources comprising computing resources and/or storage resources;

determining, by the data processing hardware, an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources;

determining, by the data processing hardware, a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time;

determining, by the data processing hardware, that the job request is capable of being executed for the amount of time; and

allocating, by the data processing hardware, resources from the pool of resources to the client device according to the resource allocation, the client device configured to execute the job request on the allocated resources over the amount of time;

managing, by the data processing hardware, a resource budget for each user of a plurality users of the pool of resources;

determining, by the data processing hardware, queuing for the plurality of users based on resource commitments of the plurality of users and the resource budget for the each user of the plurality of users,

wherein the determining an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources comprises:

estimating resource commitments for each of the plurality of users based on the historical resource usage of the pool, and

estimating the available resources from the pool of resources from the estimated resource commitments.

14. The computer-implemented method of claim **13**, wherein determining the resource allocation from the pool of resources comprises, for each unit of time, determining the resource allocation for the job request based on the estimated available resources for the each unit of time.

15. A computer-implemented method for managing a pool of resources, the method comprising:

receiving, at data processing hardware in communication with the pool of resources via a network, a job request for execution on the pool of resources, the job request received from a client device in communication with the data processing hardware and the pool of resources via the network, the pool of resources comprising computing resources and/or storage resources;

determining, by the data processing hardware, an amount of time to be utilized for executing the job request based on available resources from the pool of resources and historical resource usage of the pool of resources;

determining, by the data processing hardware, a resource allocation from the pool of resources, wherein the resource allocation spreads the job request over the amount of time;

determining, by the data processing hardware, that the job request is capable of being executed for the amount of time; and

allocating, by the data processing hardware, resources from the pool of resources to the client device according to the resource allocation, the client device configured to execute the job request on the allocated resources over the amount of time;

managing, by the data processing hardware, a resource budget for each user of a plurality users of the pool of resources;

determining, by the data processing hardware, queuing for the plurality of users based on resource commitments of the plurality of users and the resource budget for the each user of the plurality of users, wherein determining the resource allocation from the pool of resources comprises:

for each unit of time, estimating the queuing of the user from the plurality of users associated with the job request; and

estimating resource commitments of users from the plurality of users ahead of the user associated with the job request in the queue.

* * * * *